Security Architecture and Design Documentation Guidance

DEVELOPMENT REPRESENTATION DOCUMENTATION (DRD)

Version 1.6

Prepared by HR CDS TT

23 June 2011

REVISION HISTORY

Name	Date	Reason For Changes	Version
HR CDS TT	24 May 2010	Document creation	1.0
HR CDS TT	16 September 2010	Review and update by Tiger Team	1.1
HR CDS TT	16 September 2010	Review and update by Tiger Team	1.2
HR CDS TT	13 January 2011	Review and update by Tiger Team	1.3
HR CDS TT	3 March 2011	Review and update by Tiger Team	1.4
HR CDS TT	19 April 2011	Review and update by Tiger Team	1.5
HR CDS TT	23 June 2011	Update by Tiger Team	1.6

ACRONYMS AND DEFINITIONS

Acronym	Definition
CCA	Covert Channel Analysis
CDS	Cross Domain Solution
DRD	Development Representation Documentation
DTLS	Descriptive Top-Level Specification
FTLS	Formal Top-Level Specification
HLD	High Level Design
LLD	Low Level Design
SFS	Security Functional Specification
SP	Security Policy

INTRODUCTION

The Development Representation Documentation (DRD) is a document set that must address the following topic areas (as appropriate for the desired robustness level):

1. security problem,
2. security objectives,
3. security policy,
4. security architecture,
5. security requirements,
6. security functional specification (SFS),
7. high level design (HLD),
8. low level design (LLD),
9. system security policy model,
10. formal top level specification (FTLS),
11. descriptive top level specification (DTLS)
12. covert channel analysis,

This documentation set provides information about the system's design and directs the implementation. It provides evidence of the effectiveness of the design and implementation and demonstrates the following properties[1]:

1. The security functionality addresses the intended security problem.
2. The security functionality is implemented correctly and is sufficiently resistant to corruption and bypass.

It should be noted that both properties need to be realized. Trust in system security may increase as confidence in these properties increase.

The paradigm for the documentation is one of design decomposition. Figures 1 through 3 depict the decomposition for Medium, Medium-High and High robustness, indicating the relationships among the various representations and the security objectives they are intended to address.

Each component of the design decomposition (e.g., functional specification, high and low level design, and implementation representation) defines an instantiation of the design at a specific level of detail.

The connectors shown in the figures represent the mappings necessary to show derivation and correspondence between components. The structure of the mapping is not prescribed, however must be described and included in the documentation set.

[1] Suitability indicates the appropriateness of the functionality to address the security problem. Implementation correctness and resistance to corruption and bypass address the assurance requirements of the functionality.

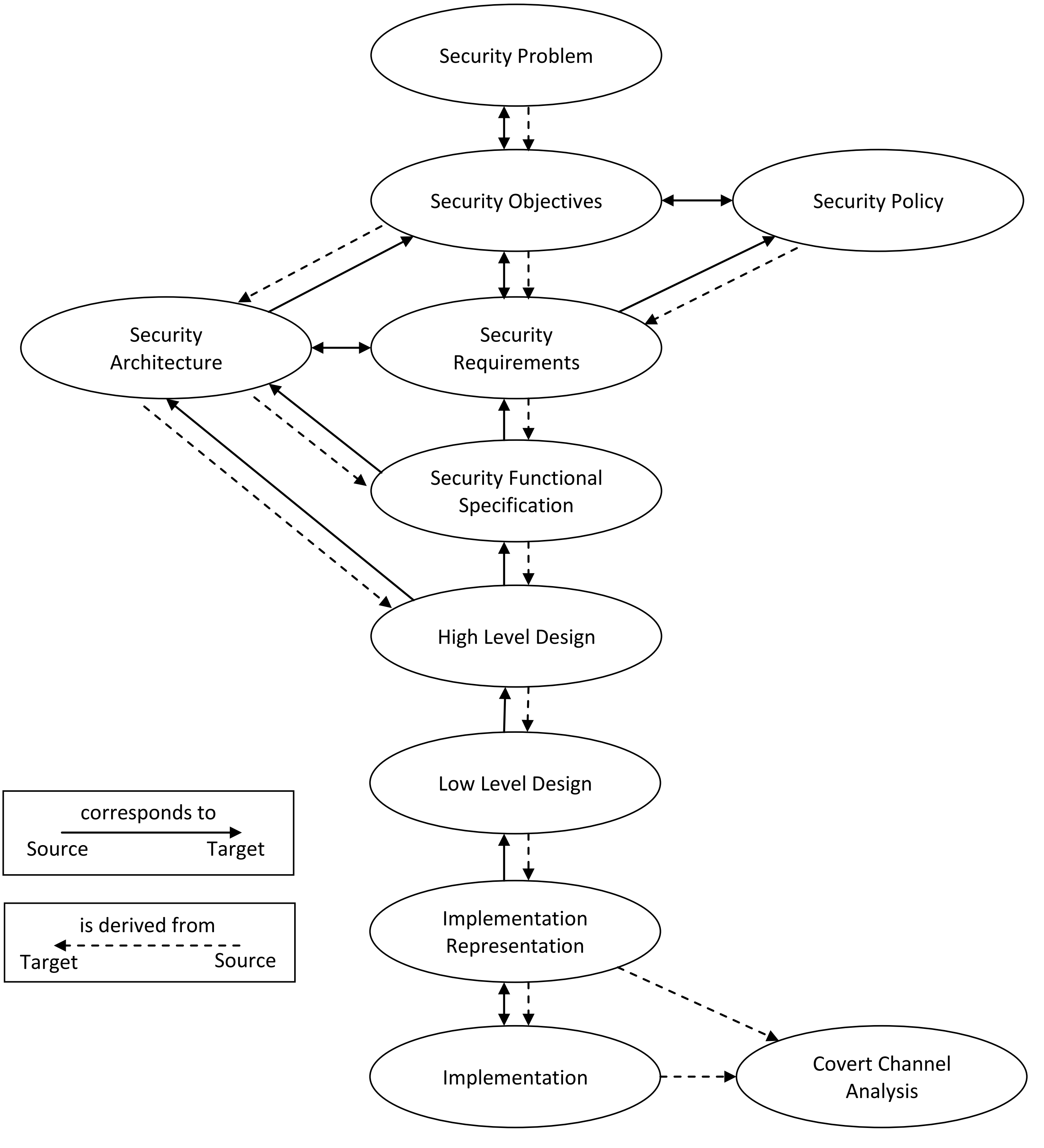

Figure 1 – Documentation Paradigm for Medium Robustness

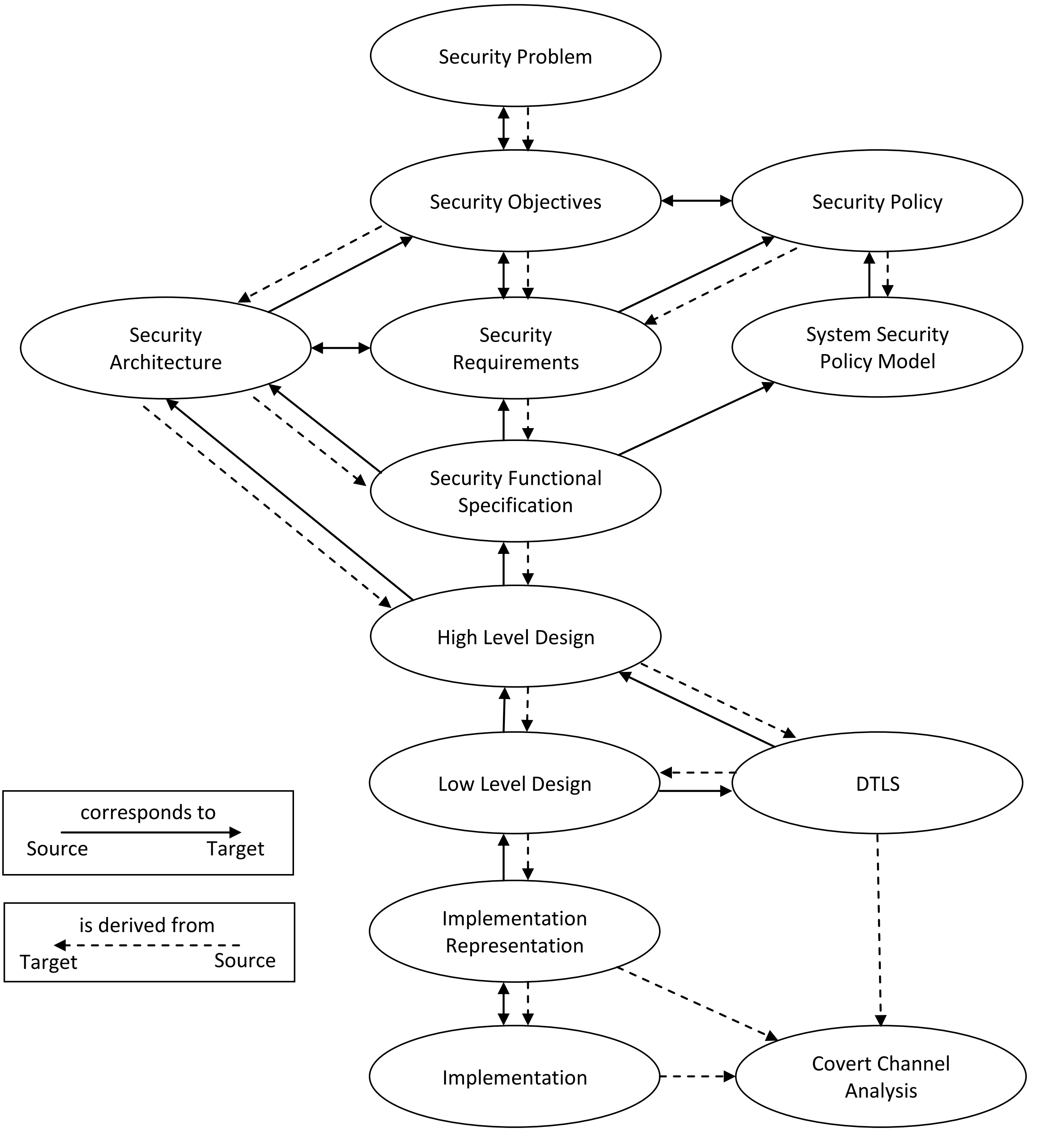

Figure 2 – Documentation Paradigm for Medium-High Robustness

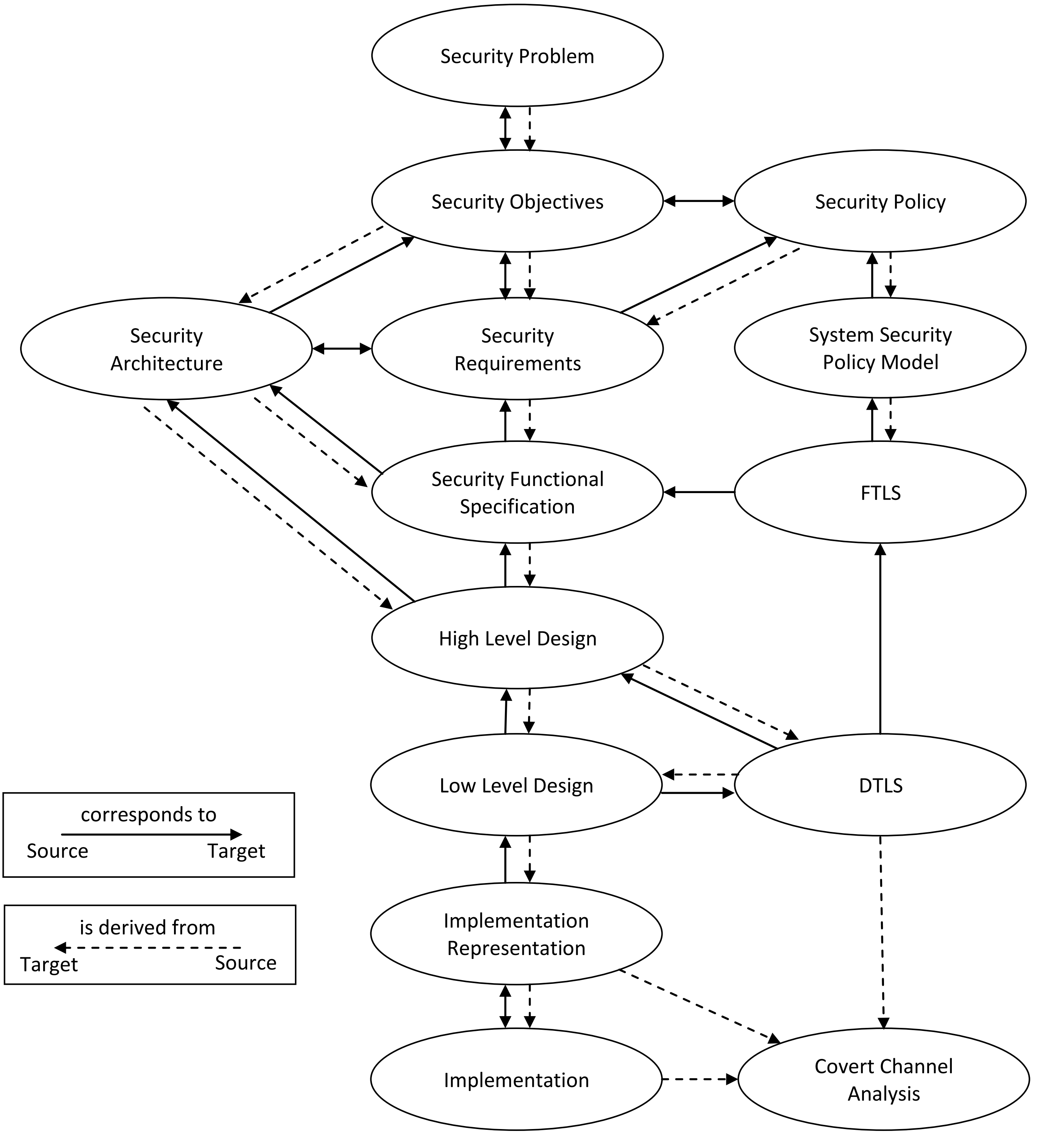

Figure 3 – Documentation Paradigm for High Robustness

- Security Problem: A concise statement of the organizational policies and procedures that the system has to implement, including the functional goals of the system, a description of the operational environment, the levels of data, and the acceptable level of residual risk.
- Security Objective: A statement of the intent to satisfy identified organization security policies and/or assumptions and to counter identified threats. A concise statement of the intended response to the security problem.
- Security Policy: A set of rules that regulate how resources are managed, protected, and distributed within a system that are further expressed by the security requirements.
- Security Architecture: A description of how the security principles (e.g., separation, isolation, least privilege, encapsulation, layering and modularity) are realized by the security design. An informal description of the overall design of a system that delineates each of the protection mechanisms employed. A combination of the formal and informal techniques used to show that the security mechanisms are adequate to enforce the security policy.
- Security Requirements: A clear, unambiguous, and well-defined description of the expected security behavior of the system.
- Security Functional Specification: A description of the system security interfaces and how they function. This consists of all means by which external entities (or subjects in the system but outside of the system's security functions) supply data to the system security functions, receive data from the system security functions, and invoke services from the system security functions.
- High Level Design: A description of the security functions in terms of major structural units (i.e. subsystems) and relates these units to the functions that they provide.
- Low Level Design: Provides more granular design information for each subsystem, including hardware, software, services and interfaces. It provides sufficient design detail to enable the production of the implementation representation.
- System Security Policy Model: Identifies the entities to be protected, who/what is allowed to access those entities, under what conditions, and in what ways. Depending upon the level of robustness required, the model may be expressed in a formal or semiformal manner. A formal representation of the model captures the entities being modeled and their relationships in a mathematically precise manner. Also, the security objectives are stated as theorems that are proven as part of the modeling process. A semiformal representation of the model captures the entities being modeled and their relationships in a natural language with precise semantics.
- Formal Top Level Specification (FTLS): A formal high-level description of the system that is used to prove the policy is satisfied.
- Descriptive Top Level Specification (DTLS): Completely and accurately describes the system in terms of its assumptions, assertions and error messages.
- Covert Channel Analysis: Identifies and potentially measures information flows in violation of the security policy.
- Implementation representation: The detailed internal workings of the security functions. This may be software source code, firmware source code, hardware diagrams and/or chip specifications.
- Implementation: The executable instantiation of the system.

DISCUSSION

The system security functionality consists of all parts of the system that have to be relied upon for enforcement of the system security policy. The security functions includes both functions that directly enforce the system security policy and also those functions that, while not directly enforcing the system security policy, contribute to the enforcement of the system security policy in a more indirect manner. As an example, functions that have the capability to cause the system security policy to be violated or portions of the system that are invoked on system start-up that are responsible for putting the system in its initial secure state.

The over-riding notion for the development representation documentation is that, as more information becomes available, greater assurance can be obtained that the security functions are correctly implemented, cannot be compromised, and cannot be bypassed. This is done through the verification that the set of development documentation is correct and consistent. This documentation provides information that can be used to ensure that the testing activities (both functional and penetration testing) are comprehensive.

It is generally the case for CDSs that there are portions of the security functions that deserve more intense examination than other portions of the security functions.

Functions are considered to be "security enforcing" if they directly implement a portion of the system security policy. Security enforcing functions make the decisions regarding system operations with respect to the system security policy.

Functions are considered to be "security supporting" if they are trusted to preserve the correctness of the system security policy by operating without error. Such functionality enables or "carries out" the decisions of the security enforcing functions. Domain separation, process isolation, resource encapsulation, and memory management are all examples of security supporting functionality that enable the security enforcing functions. Input/output operations are examples of the security supporting functionality that "carries out" the decisions of the security enforcing functions.

Functions are considered to be "security non-interfering"[2] if they do not enforce or support any aspect of the system security policy but, due to their presence inside the security boundary, they must be correct or they could adversely affect the correct implementation of the security policy. Security non-interfering functions have no role in implementing the security policy, and are likely part of the security functionality because of their environment; for example, any code running in a privileged hardware mode within an operating system. Such functions need to be considered part of the security functionality because, if compromised (or replaced by malicious code), it could compromise the correct operation of a security function by virtue of its operating in the privileged hardware mode. An example of security non-interfering functionality is a set of mathematical floating-point operations implemented in kernel mode for speed considerations.

All three of the above types of functions (i.e., security enforcing, security supporting, and security non-interfering) are "security relevant".

The architecture documentation provides a description of the system architecture based on security principles and how they combine to satisfy the system security policy. These architectural artifacts are at least as important, if not more important, than the security functions. If these artifacts are not present, it will likely lead to a failure of the mechanisms implementing the security functions.

The difference in analysis of the implementation of the security functionality and of the implementation of the security principles is that the functionality is more or less directly visible

---

[2] This is not to be confused with the Non-Interference security model first articulated by Goguen and Meseguer.

and relatively easy to test, while the security principles require varying degrees of analysis. Further, the depth of analysis possible for these principles will vary depending on the design of the system.

The amount and structure of the design documentation will depend on the complexity of the system and the number of security requirements; in general, a very complex system with a large number of security requirements will require more design documentation than a very simple system implementing only a few security requirements.

PRESENTATION

The information about each DRD topic area is required, rather than a particular document set structure; therefore, it is not necessary for every DRD topic area to be in a separate document[3]. Indeed, it may be the case that a single document meets the requirements for more than one topic area. In cases where multiple topic areas are combined within a single document, the developer must indicate which portions of the document apply to which topic areas.

FORMALITY

Three types of specification style are mandated by this DRD: informal, semiformal, and formal. The functional specification and system design documentation are always written in either informal or semiformal style. A semiformal style reduces the ambiguity in these documents over an informal presentation. A formal specification may also be required in addition to the semi-formal presentation; the value is that a description of the security functions in more than one way will add increased assurance that the security functions have been completely and accurately specified.

An informal specification is written as prose in natural language (i.e., English). An informal specification is not subject to any notational or special restrictions other than those required as ordinary conventions for that language (e.g. grammar and syntax). While no notational restrictions apply, the informal specification is also required to provide defined meanings for terms that are used in a context other than that accepted by normal usage.

The difference between semiformal and informal documents is more than a matter of formatting and presentation: semiformal notation includes such things as an explicit glossary of terms, a standardized presentation format, etc. The presentation should use terms consistently and may also use more structured languages/diagrams (e.g. data-flow diagrams, state transition diagrams, entity-relationship diagrams, data structure diagrams, and process or program structure diagrams). The glossary identifies the words that are being used in a precise and consistent manner; similarly, the standardized format implies that extreme care has been taken in methodically preparing the document in a manner that maximizes clarity. It should be noted that fundamentally different portions of the system might have different semiformal notation conventions and presentation styles (in such cases, a mapping must be provided).

A formal specification is written in a notation based upon well-established mathematical concepts, and is typically accompanied by supporting explanatory (informal) prose. These mathematical concepts are used to define the syntax and semantics of the notation and the proof rules that support logical reasoning. The syntactic and semantic rules supporting a formal notation should define how to recognize constructs unambiguously and determine their meaning. There needs to be evidence that it is impossible to derive contradictions, and all rules supporting the notation need to be defined or referenced.

---

[3] Documentation may take a number of forms. Any tools (including training and support) necessary to read and understand the documention shall be provided.

The formality of the presentation of the DRD topic areas vary from Informal (I), Semiformal (SF) and Formal (F) depending on a number of factors. Some topic areas, such as the problem statement need to be generally understood and therefore are presented informally. Higher levels of robustness may require the presentation to be in a more formal manner. Table 1 summarizes the presentation formality with the corresponding levels of robustness. The DRD topic areas and assurance levels not listed in Table 1 require informal presentation.

Table 1 – DRD topic areas and corresponding formality based on levels of robustness.

	Medium	Medium High	High
Security Policy	I	SF	SF
Policy Model	SF	F	F
FTLS	N/A	N/A	F
DTLS	N/A	I	I
HLD	I	SF	SF
LLD	I	I + SF	SF + F
Security Architecture	I	SF	SF
CCA[4]	I	SF	F

[4] Further guidance on CCA requirements and presentation is found in the CCA guidance document.